ЗА УСВОЯВАНЕТО НА АЛГОРИТМИ В ОБУЧЕНИЕТО ПО ПРОГРАМИРАНЕ[1]

гл.ас. Тодорка Терзиева

Пловдивски университет „Паисий Хилендарски”
Факултет по математика и информатика
4003 Пловдив, бул. „България” 236
dora@uni-plovdiv.bg

Абстракт: *Разработването на алгоритми е един от най-трудните етапи при решаване за задачи с помощта на компютър. В първоначалния етап на обучението по програмиране е целесъобразно да не се преминава директно към някакъв алгоритмичен език за програмиране, а да се акцентира върху разбиране и разработването на алгоритъм. Предложена е методика за обучение на алгоритми чрез постепенно нарастване нивото на сложност.*

Ключови думи*: алгоритъм, алгоритмично мислене*

1. Въведение

Обучението по алгоритмизация и програмирането, като начин за реализация на алгоритмите, са от основните компоненти на училищния курс по информатика. Една от целите на обучението по информатика е „запознаване с понятието алгоритъм, начините на описание на алгоритми, основни алгоритмични конструкции и структури от данни, решаване на несложни задачи чрез средствата на алгоритмизацията” [1].

Известно е, че повечето обучаеми не притежават умения за разработване на алгоритъм за решаване на сложна задача. Особени трудности се срещат при необходимост от обобщен анализ на данните и формализация, липса на умение решението за една задача да се представи със средствата на формалните езици. Всеки алгоритъм се състои от логически свързани дискретни части (елементарни действия), които от гледна точка на системния подход се разглеждат като единно цяло [3]. Вследствие на това, нивото на развитие на алгоритмичните способности на обучаемите е тясно свързано с нивото на абстрактност на мисленето и със способността за логически умозаключения [2].

Голяма част от обучаемите могат да решават задачи, които са основани на решавани преди това аналогични примери, т.е. липсва способност за продуктивно творчество. Изучаването на нови теми в този случай може да започне с предложение към обучаемите *сами да влязат в ролята на изпълнители на зададен по различни начини алгоритъм*. Развитие на умението за разбиране и изпълнение на разработени алгоритми (словесни, блок-схеми или програми) е ефективно средство за формиране на алгоритмичен стил на мислене.

Разработването на алгоритми трябва да се осъществява чрез постепенно нарастване нивото на сложност: реализация на линеен алгоритъм, разклонен алгоритъм, цикъл с параметър, детерминиран цикъл. След това може да се пристъпи към задачи за обработка на структурирани абстрактни структури от данни – масиви, низове, записи, множества и др. При това е необходимо обучение, насочено към създаване и развитие на умения за самостоятелно оценяване правилността на разработената програма на основата на анализ на получените резултати.

2. Примерни задачи

Основен етап в програмирането е разработването на алгоритми. Това е един от най-трудните етапи при решаване за задачи с помощта на компютър. В първоначалния етап на обучението по програмиране е целесъобразно да не се преминава директно към някакъв алгоритмичен език за програмиране, а да се акцентира върху разработването на алгоритъм. След това може да се премине към произволен език за програмиране.

Разработването на един по-сложен проблем може да стане постепенно, като се започне с едно елементарно действие и постепенно се въвеждат различни условия, контрол на входни данни, анализ на изходни резултати и други.

За илюстрация на този подход ще разгледаме няколко задачи.

Задача 1. Да се намери най-голямото от 3 цели числа.

[1] Тази работа е подпомогната по проект ИС–М–4 на поделение „Научна и приложна дейност” при Пловдивския университет „Паисий Хилендарски”.

Словесен алгоритъм (Фиг.1)

Вход: Три променливи **a**, **b**, **c**. **Изход:** Максималната стойност – **max**.

Стъпка 1. Въвеждане на входни данни;

Стъпка 2. Приема се, че максималната стойност е първата **max=a**.

Стъпка 3. Проверява се дали **max<b**. Ако отговорът е ***„да"*** се изпълнява *стъпка 4.* Ако отговорът е ***„не"*** се преминава към *стъпка 5.*

Стъпка 4. Променливата **max** присвоява стойността на **b**, **max=b**.

Стъпка 5. Проверява се дали **max<c**. Ако отговорът е ***„да"*** се изпълнява *стъпка 6.* Ако отговорът е ***„не"*** се преминава към *стъпка 7.*

Стъпка 6. Променливата **max** присвоява стойността на **c**, **max=c.**

Стъпка 7. Извеждане (отпечатване) стойността на **max** като резултат.

Задача 2. Да се намери най-голямото от 10 на брой цели числа.

Очевидно основните действия, които трябва да се извършват, са сравнение на две стойности и присвояване на стойност. Какви са специфичните проблеми, който трябва да се решат в сравнение с предишната задача?

- Посочените елементарни действия трябва да се повтарят многократно;
- Избор на структура от данни, която ще се използва за съхранение на тези 10 стойности.

Въвежда се понятието *цикъл*, като средство за реализиране на действие или група от действия, които се повтарят многократно. Необходимо е да се прецени кой от трите вида цикли е удобно да се използва.

Вторият от проблемите е свързан със структурата от данни. Възможно е решение с използване само на една променлива за съхранение на въвежданите числа и съответно една за съхранение на максималната стойност. При този вариант (Фиг. 2), обаче, въведените стойности от клавиатура се запазват временно, използват се за сравнение с текущата стойност на *max*, след което се въвежда нова стойност от клавиатура, която се присвоява на променливата *a*. Ако искаме да запазим стойностите на всяко едно от въведените числа, да ги изведем и съответно да ги използваме за по-нататъшни обработки, е необходима друга структура от данни. При дефиниране на една променлива *a*, едни и същи клетки от паметта се използват за съхранение на всяко следващо въведено число. Ако изведем стойността на променливата *a*, след приключване на програмата тя ще съдържа последната въведена стойност.

За съхранение на елементи от един и същи тип се въвежда абстрактен тип данни – едномерен масив. Тъй като елементите на масива са разположени последователно в паметта, то *arr[i]* ще сочи i-тият елемент на масива, следователно достъпът до всеки елемент е чрез името на масива и неговия индекс. Памет за масивите се заделя по време на компилация, поради което техните размери трябва да бъдат константи или константни изрази. Тъй като масивът трябва да съдържа 10 елемента, т.е. броят на повторенията на елементарните действия (сравнение и присвояване) не зависи от някакво условие, а е известен предварително, е целесъобразно да се използва детерминиран цикъл с параметър.

```
// фигура1
#include<iostream>
using namespace std;

void main()
{
        int a,b,c,max;
       cout<<"a b c "<<endl;
        cin>>a>>b>>c;
max=a;
      if (max<b) max=b;
      if (max<c) max=c;
        cout<<"max="<<max<<endl;
system "pause";
}
```

```
// фигура 2
#include<iostream>
using namespace std;

void main()
{   int a,i,max;
    cout<<"a="; cin>>a;
    i=1;
   max=a;
   do
  {   cout<<"a=";
      cin>>a;
   If (max<a) max=a;
     i=i+1;
   } while (i<=10);
```

	cout<<"max="<<max<<endl; }
Задача 1.	**Задача 2.**

Задача 2. Словесен алгоритъм – (Фиг.2)

Вход: Променливи **a**, **i**. **Изход:** Максималната стойност – **max.**

Стъпка 1. Въвеждане на стойност на променливата **a** от клавиатура и **i:=1** ;

Стъпка 2. Приема се, че максималната стойност е първата **max=a**.

Стъпка 3. Въвеждане на стойност на променливата **a** от клавиатура.

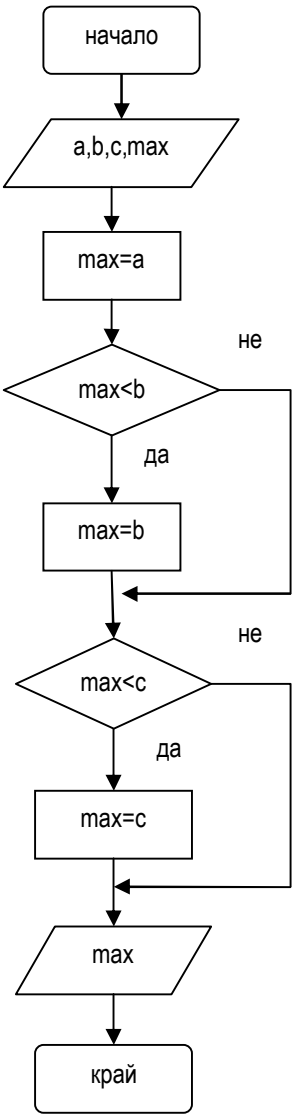
Стъпка 4. Проверява се дали **max<a**. Ако отговорът е ***„да"***, се изпълнява *стъпка 5.* Ако отговорът е ***„не"***, се преминава към *стъпка 6.*

Стъпка 5. Променливата **max** присвоява стойността на **a**, **max=a**.

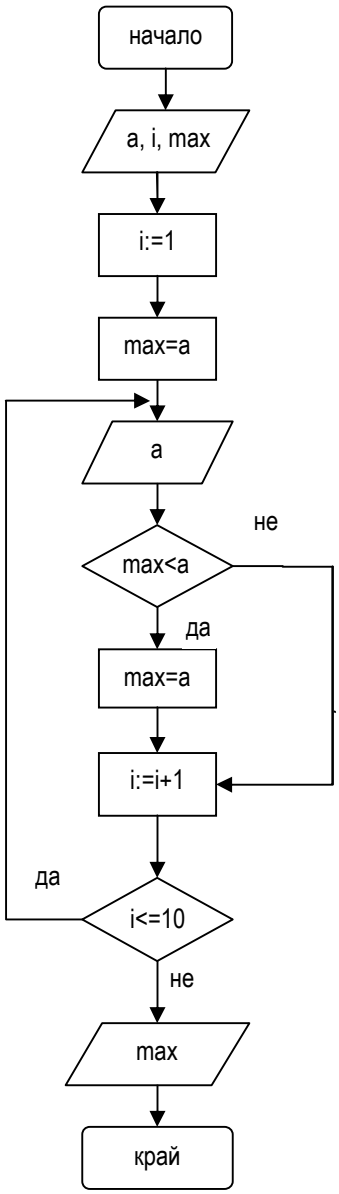
Стъпка 6. Стойността на променливата **i** се увеличава с 1, **i:=i+1**.

Стъпка 7. Проверява се дали **i<=10**. Ако отговорът е ***„да"***, се изпълнява *стъпка 3*. Ако отговорът е ***„не"***, се преминава към *стъпка 8.*

Стъпка 8. Извеждане /отпечатване/ стойността на **max** като резултат.



Фиг. 1 Задача 1



Фиг. 2 Задача 2

```
  #include<iostream>
using namespace std;
void main()
{
   int arr[11],i,max;
   for(i=1;i<=10;i++)
   {   cout<<"arr["<<i<<"]=";
       cin>>arr[i];
   }
    max=arr[1];
    for (i=2;i<=10;i++)
    if (max<arr[i]) max=arr[i];
    cout<<"max="<<max<<endl;
}
```

```
program max_array;
type masiv=array[1..10] of integer;
var
    arr:masiv;
    i,max:integer;
begin
for i:=1 to 10 do
  begin
    write('arr[',i,']=');
    read(arr[i]);
  end;
max:=arr[1];
for i:=2 to 10 do
 if max<arr[i] then max:=arr[i];
writeln ('max=',max);
end.
```

Задача 2. Програма на C++ с масив	**Задача 2. Програма на Паскал с масив**

При реализация на този алгоритъм на C++ се получава един трудно откриваем проблем. Забелязва се, че в програмата се дефинира едномерен масив от цели числа тип (int arr[11]), а броят на въведените стойности трябва да бъде 10. Това е един често срещан и труден за отстраняване проблем от начинаещи в обучението по програмиране. Ако алгоритъмът се реализира на Паскал, този проблем не съществува.

Обикновено на обучаемите се обяснява, че индексът на масива в C++ започва от 0 до (n-1), където n е брой елементи на масива. Разбирането за адресиране и разположението на първи елемент на масива с индекс arr[0], трудно се асоциира с пореден номер 1 на елемент на масива.

Ако освен максималната стойност трябва да се изведе и пореден номер на елемент, то поредният номер трябва да бъде (n+1), при реализация на алгоритъма на C. В решението се декларира брой на елементи на едномерния масив 11 – (int arr[11]), като по този начин индексът на елементите започва с номер 1 и се разполага в паметта на адреса на вторият елемент. По този начин първият елемент на масива остава неизползваем, но при декларацията е необходимо да се резервират (n+1) на брой елементи.

По-естествена е реализацията на този алгоритъм на Паскал, където този проблем не съществува.

На базата на последното решение може да се обобщи задачата и да се параметризира.

Задача 3. Да се намери най-голямото от произволен брой, въведени от клавиатурата, цели числа.

```
#include<iostream>
using namespace std;
void main()
{
   const N=100;
   int arr[N],i,n, max;
   cout<<"n="; cin>>n;
   for(i=0;i<n;i++)
   {   cout<<"arr["<<i<<"]=";
       cin>>arr[i];
    }
    max=arr[0];
    for (i=1;i<n;i++)
     if (max<arr[i])
        max=arr[i];
     cout<<"max="<<max<<endl;
}
```

Задача 3. Решение на C++.

```
program max_array2;
 type masiv=array[1..100] of integer;
  var     arr:masiv;
          i,n,max:integer;
begin
       write('n='); read(n);
       for i:=1 to n do
          begin
             write('arr[',i,']=');
             read(arr[i]);
          end;
       max:=arr[1];
       for i:=2 to n do
          if max<arr[i] then max:=arr[i];
       writeln ('max=',max);
       readln
                    end.
```

Задача 3. Решение на Паскал

След реализиране на тези алгоритми може да се предложат за обсъждане и реализация от обучаемите следните задачи:

Задача 4. Да се намери най-голямото (най-малкото) от произволен брой, въведени от клавиатурата различни цели числа и да се изведе неговият индекс.

Задача 5. Да се намери най-голямото (най-малкото) от произволен брой, въведени от клавиатурата цели числа и да се изведат индексите на елементите със същата стойност.

Задача 6. Да се намери най-малкото от *N* на брой цели числа, които се делят на 3 без остатък.

Задача 7. Да се намери сумата и произведението на най-малкото и най-голямото от *N* на брой реални числа, (5<=N<=15).

Следващата тема е по-сложна и е свързана с изучаване на алгоритми за сортиране.

Задача 8. Да се въведат *N* на брой реални числа, да се подредят (сортират) във възходящ (низходящ ред) по метода на пряка селекция (пряк избор) и да се изведат подредените стойности. (10<=N<=20).

3. Изводи

Чрез използване на средствата и методите на програмирането се стимулира развитието на способностите на алгоритмично мислене, формират се умения за:

- анализ на входните данни, ясно разграничаване на това, което е дадено и това, което се търси;
- изпълнение на зададен алгоритъм;
- анализиране на готови програми – откриване на синтактични и семантични грешки;
- проверка за изпълнение на всички основни свойства на алгоритмите;
- откриване на основните действия, които са подходящи за решение на даден проблем;
- анализ на получените резултати;
- разработване на алгоритъм с използване на съответния формален език;
- избор на най-ефективно решение.

Литература

[1]. Асенова, П., П. Михнев, Информатиката и информационните технологии като част от общото образование, Информатиката и информационните технологии в обучението (Опит и тенденции). МОН и НИИО, С., 1997.

[2]. Amorim, C., BrazilBeyond Algorithmic Thinking: An Old New Challenge for Science Education. Universidade do Estado da Bahia, Salvador, BA.

[3]. Iliev A., G. Hristozov. Software Environment for Dynamic Models Building, Conference Proceedings of 30th International Conference - Information and Communication Technologies and Programming (ICT&P), Sofia, Bulgaria,
June 23-24, 2005, 125-132.